Path Optimization for Mobile Robots using Genetic Algorithms

Fernando Martínez Santa, Fredy H. Martínez Sarmiento, Holman Montiel Ariza
Universidad Distrital
Francisco José de Caldas
Bogotá, Colombia

***Abstract*—This article proposes a path planning strategy for mobile robots based on image processing, the visibility graphs technique, and genetic algorithms as searching/optimization tool. This proposal pretends to improve the overall execution time of the path planning strategy against other ones that use visibility graphs with other searching algorithms. The global algorithm starts from a binary image of the robot environment, where the obstacles are represented in white over a black background. After that four *keypoints* are calculated for each obstacle by applying some image processing algorithms and geometric measurements. Based on the obtained *keypoints*, a visibility graph is generated, connecting all of these along with the starting point and the ending point, as well as avoiding collisions with the obstacles taking into account a safety distance calculated by means of using an image dilation operation. Finally, a genetic algorithm is used to optimize a valid path from the start to the end passing through the navigation network created by the visibility graph. This implementation was developed using *Python* programming language and some modules for working with image processing ang genetic algorithms. After several tests, the proposed strategy shows execution times similar to other tested algorithms, which validates its use on applications with a limited number of obstacles presented in the environment and low-medium resolution images.**

Keywords*—*Optimization; path planning; genetic algorithms; visibility graphs

I. Introduction

Today more than ever, robotics is part of the daily life in most of the world specially in big cities where the automation and smart stuff is everywhere. Mobile robots area has been widely researched not only in its mechanical design and locomotion type but in its motion planning [1], [2] in applications such as movement in indoor environments [3], obstacle avoidance [4], navigation in complex mazes [5], path planning [6], [7], and some times using open source robotics software [8]. Researching areas like UAVs (Unmanned Aerial Vehicles) and self-driving or autonomous vehicles have maintained the interest on one of the most important issues for mobile robots, the path planning. In this area, one of the most used algorithm has been the visibility graphs [9] supported by image processing algorithms [10]. These visibility graphs generates a high dense network of possible paths through the some navigation keypoints obtained from the obstacles image that is related with the navigation scene to solve by the mobile robot. This path network includes also both the starting point and the ending point, then a valid and short path has to be found from the start to the end passing through some segments of the connection network, for that reason it is necessary to apply a decision or optimization algorithm [11] to choose the best path inside on that network. A lot of different optimization algorithms has been used for the path planning issue such as ant colony optimization [12], [13], particle swarm for mobile robots [14], [15], [16], [17], chaotic particle swarm [18] particle swarm for manipulators [19], brain storm optimization [20], Fuzzy-Wind Driven algorithm [21], rapidly-exploring trees [22], gray wolf algorithm [23] among others.

The Genetic Algorithms *GAs* are searching and optimization methods based on the natural selection process and the genetic operations involved in it, these ones have been used for solving problems in a lot of different engineering areas including robotics and of course path planning [24], [25], [26] and other variations such as the Taxi carpooling algorithm [27].

Therefore, this research aims to propose a path planning strategy combining both the visibility graphs method using image processing algorithms and genetic algorithms to optimize and obtain the best path, improving the overall execution time respect other related works. All of this strategy is proposed to be solved by means of using free software in this case all of the algorithms will be implemented on *Python* language using modules such as: *scikit-image* and *geneticalgorithm2*.

The paper is organized as follows: Section 2 presents the methodology proposed to find the optimal path for mobile robots using Genetic Algorithms (GAs), describing all processes to identify the obstacles image by capturing the image of navigating environment; going through the *keypoints* obtaining, then generating visibility graph, and subsequently selecting path optimization using GAs. Section 3 presents the results of implementing the path planning strategy in Python 3 language and testing in different navigation environments to evaluate overall execution time. Finally, Section 4 presents the conclusions about this research's main ideas, including possible future jobs.

II. Methodology

The path planning strategy for mobile robots proposed in this article is based on image capture of the navigating environment in which the robot is involved [28], [29], where the obstacles are perfectly differentiated from the void room. First step is the calculation of some *keypoints* for each obstacle in the scene in order to reduce the amount of information to work with by the planning algorithm. After that, all the possible paths from the start to the end are calculated, all of them passing through the previously detected *keypoints*, producing a

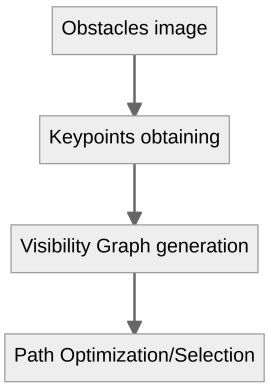

Fig. 1. General Algorithm Pipeline.

highly dense path network. Finally, the optimization algorithm [30] is applied to the path network in order to obtain the shortest path. The complete process can be sumarized through the flow chart shown in Fig. 1, where it starts from a binary image of the environment (a black background and white obstacles), then some *keypoints* by each obstacle are calculated, after that a visibility graph is generated and finally only one path is selected. In the next sections, each step of the process is explained in detail.

A. Obstacles Image

The proposed strategy starts with a Black & White image of the scene with the obstacles [31], this binary image has a black background and the obstacles are represented in white as the example shown in Fig. 2. This image can be obtained from a camera located at the top of the robot environment and after turned into binary by means of applying an image threshold operation.

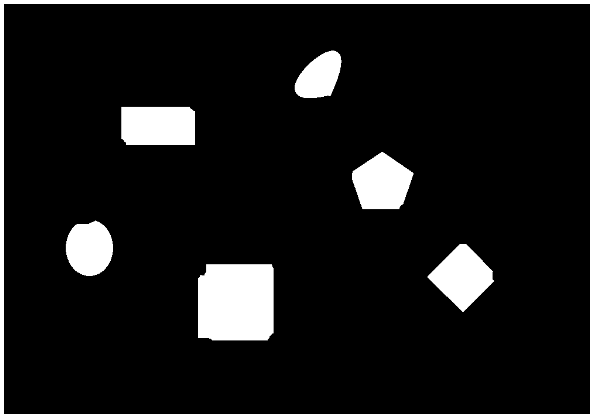

Fig. 2. Starting Scene Binary Image.

B. Keypoints Obtaining

The *keypoints* obtaining step is supported by some digital image algorithms, first the binary image of the obstacles is dilated in order to expand the obstacles border, applying the correspondent image morphology operation. After, the main two axis and the centroid of each dilated obstacle are calculated, this is done labeling and measuring each separated region in the binary image (obstacles). Finally, four *keypoints* per obstacle are calculated.

1) Image Dilation: In this proposal, the navigation *keypoints* are based on the obstacles borders, but this takes into account a safe distance between these ones and the robot [32]. That distance is calculated from to the maximum radius of the robot according to the eq. 1 and correspons to the dilation radius r_d.

$$r_d = \lfloor r_m + \Delta r \rfloor \tag{1}$$

Where r_m is the maximum radius of the robot and Δr is a radius tolerance defined at 10% in this case. Finally, r_d has to be an integer and it is represented in pixel units.

Once the dilation radius r_d is obtained, the morphology dilation operation is performed on the obstacles image by means of applying a $2D$ convolution between the original binary image and a square shape as wide as r_d. The result is shown in Fig. 3, where the obstacles are the same as the binary image (see Fig. 2), but their area is expanded because of the dilation operation. This operation allows assuming obstacles with major areas to avoid future collisions due to the maximum radius of the mobile robot.

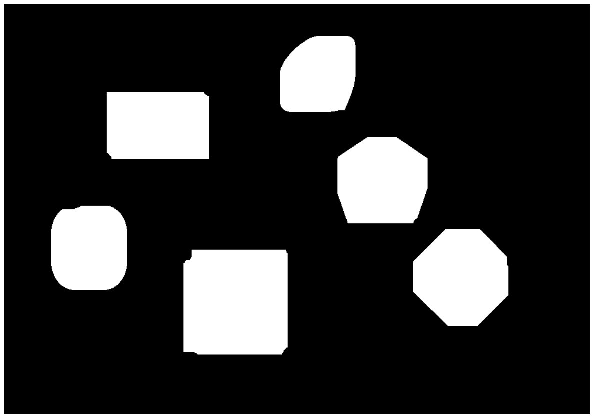

Fig. 3. Dilated Obstacles Image.

2) Obstacle keypoints Computing: After dilating the obstacles image, each obstacle is labeling and measured in order to find its centroid and its two main axis, from this data a Δx and a Δy are calculated for each axis according to eq. 2.

$$\forall i \in I \quad : \begin{cases} \Delta x_i = \cos(\theta) \cdot \left(\frac{l_i}{2}\right) \\ \Delta y_i = \sin(\theta) \cdot \left(\frac{l_i}{2}\right) \end{cases} \tag{2}$$

Where l_i is the length of each main axis i of the obstacles axis set I and θ the orientation of the major axis detected. Finally from these deltas, the *keypoints* are calculated according to the eq. 3.

$$\forall i \in I,\ \forall j \in J \quad : \begin{cases} x_{ij} = x_0 \pm \Delta x_i \\ y_{ij} = y_0 \pm \Delta y_i \end{cases} \tag{3}$$

Where j is each of the calculated points set J for each main axis set I and (x_0, y_0) is the centroid of each obstacle. The cardinality of the sets I and J is $|I| = |J| = 2$, so for each obstacle two main axis are calculated, and for each of them two points are generated, for a total of 4 generated *keypoints* by obstacle, as shown in Fig. 4.

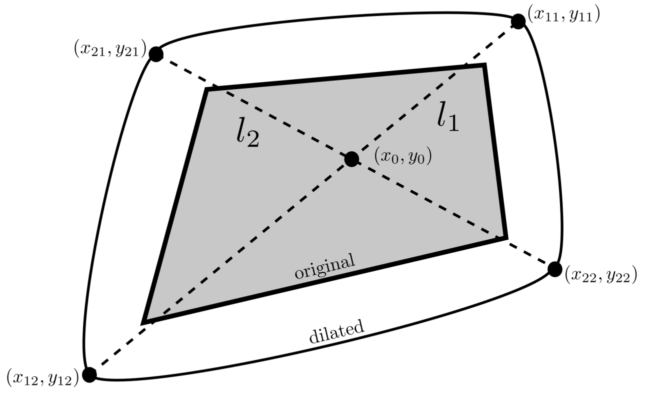

Fig. 4. Obstacle *keypoints* Computing.

The Fig. 4 shows schematically the dilation process. Having an diamond-shaped obstacle (shown in gray) in the original image, the external rounded shape (in white) represents the same obstacle after the image dilation process. The main axis are represented by dashed lines, (x_0, y_0) is the centroid and the (x_{ij}, y_{ij}) are the obtained *keypoints*. The obtained *keypoints* along with the original obstacles are shown in Fig. 5, where the starting point is at the lower-left corner and the ending point is at the right border.

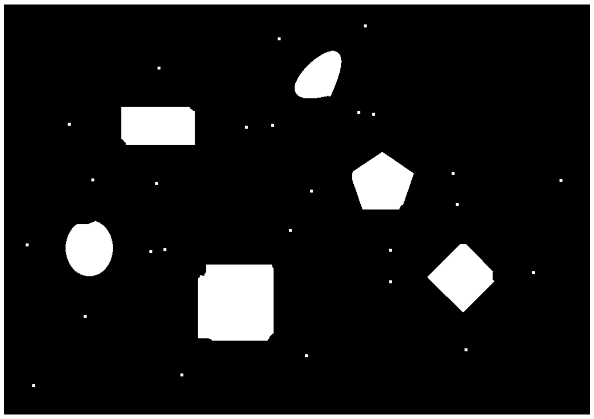

Fig. 5. Navigable *keypoints* Image.

C. Visibility Graph Generation

After generating all of the navigating *keypoints* including the starter and ending points, the visibility graph [33] is generated (see Fig. 6) connecting all of these points (drawing lines on the image) and after avoiding the crossing with the obstacles as shown in Fig. 7.

The collision avoidance between these lines and the obstacles is calculated by means of applying binary image

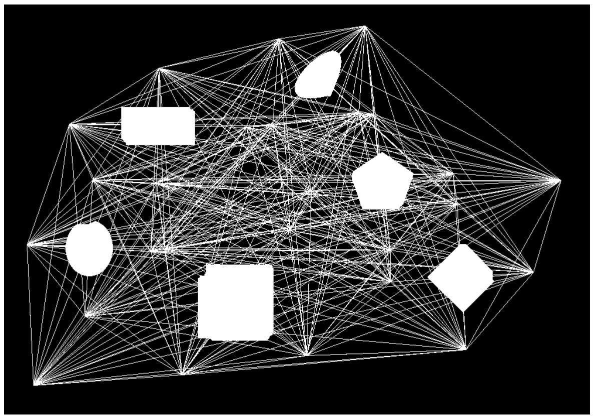

Fig. 6. Visibility Graph.

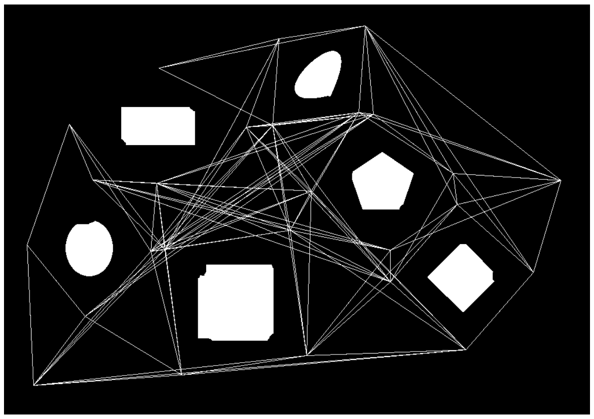

Fig. 7. Visibility Graph (Avoiding Collisions).

operations, specifically a binary *XOR* operation (exclusive disjunction) between the dilated obstacles image and a copy of the same image with the specific line drawn in *black*. If there is a collision, a black segment line will appear over an obstacle, that means that the two images will be different. Both images have to be totally equals (pixel by pixel) for validating the line, so each resultant pixel has to accomplish the eq. 4,

$$\forall i \in I, \forall j \in J \quad : A_{ij} \nLeftrightarrow B_{ij} = \mathit{False} \tag{4}$$

Where I and J for this equation, are the sets of all valid indices in both dimensions of the dilated obstacles image A and the image B which has the dilated obstacles plus the drawn line.

D. Path Optimization-Selection

For selecting the shortest possible path in the generated visibility graph, it is possible to use a lot of different selection or optimization algorithms such as the A^* algorithm [34]. In this proposal, Genetic Algorithms (GAs) are used as optimization tool [35] in order to find the shortest (and then the most efficient) path in the dense navigating network generated by the visibility graph.

1) Target Function: Once defined the complete set of *keypoints* P including the start p_s and the end p_e, it is necessary to define the *target function* to optimize $f(P)$, this depends on the cumulative distance of each segment $\overline{p_i p_{(i+1)}}$ from p_s to p_e, that accomplishes with the eq. 4. For the target function definition, it is necessary to define the solution set X as shown in eq. 5, where each x represents an index for reading the *keypoints* set P, so the set X detemine the order of a subset $P_X \subseteq P$ which is a possible solution (a short path).

$$X = \{x_0, x_1, x_2 \dots x_n\} \tag{5}$$

The number of indices of the solution set X corresponds to the number of objects in the scene, plus the inital and ending points thus $n = \textit{n-obstacles} + 2$. The number of elements of the set X is less or equal of the number of elements of the subset P_X, so $|P_X| \leq |X|$, This occurs because a P_{x_i} different from Px_n equals the ending point P_e, that is meant the path reaches the final in less steps than the maximum allowed n, so it is necessary to determine the real number of steps m. This last is possible applying the eq. 6.

$$\forall i \in \{0, 1, 2 \dots n\} \ : \ p_{x_i} = p_e \implies m = i \tag{6}$$

Once the steps m have been computed, the optimization target function is defined from the solution set X as shown in eq. 7.

$$f(X) = \sum_{i=0}^{m} |\overrightarrow{p_{x_i} p_{(x_i+1)}}| \tag{7}$$

Where $|\overrightarrow{p_{x_i} p_{(x_i+1)}}|$ is the distance of a segment between two sequential *keypoints*. This target function is subject to the first element of P_k were the starting point p_s, then it is generating the optimization restriction shown in eq. 8.

$$p_{x_0} = p_s \tag{8}$$

Additionally, as were described in the visibility graph section, the target function also has an obstacle collision restriction which can be implemented applying the eq. 4

2) Genetic Algorithm Implementation: The genetic algorithm proposed in this article for optimizing the visibility graph is setup as follows: there is no limit for the maximum number of iterations, the number of iterations without any improvement in the target function (fitness function) is set in 20, the crossover and mutation type era defined as uniform, the selection type is roulette, a 100 individuals population is defined, the rest of parameters are shown in Table I.

The *fitness function* or target function to minimize by the genetic algorithm is implemented according to the eq. 7 and specifying the restrictions (see eq. 4 and eq. 8), generation a penalty when one of them is not accomplished, that is meant $f(P)$ is carried to a maximum value $f(P)_{max}$ which corresponds to the total perimeter of the image A, then $f(P)_{max} = A_{x_{max}} * 2 + A_{y_{max}} * 2$.

TABLE I. Genetic Algorithm Parameters

Parameter	value
maximum iteration number	None
population size	100
mutation probability	0.1
elitism ratio	0.01
crossover probability	0.5
parents portion	0.3
crossover type	uniform
mutation type	uniform by center
selection type	roulette
max. iteration without improvement	20
dimension	same number of obstacles
variable type	integer
function timeout	10s

On the other hand the genome is built from the set X, taking into account that $\forall i \in \{0, 1, 2 \dots n\} : x_i \in \mathbb{E}$, the genome is simply generated in order as shown in Table II, being each gene an integer variable. No other variable different to the x_i is necessary to be appended to the genome.

TABLE II. Genome Order.

gen_0	gen_1	gen_2	gen_3	$\dots$	gen_{n-1}
x_0	x_1	x_2	x_3	$\dots$	x_{n-1}

III. Results

All of the proposed path planning strategy was implemented in *Python 3* language on a simple office laptop (whose features are described in Table III) running a *GNU/Linux* distribution. In total 10 tests were done over different navigation environments, all of them with the 6 obstacles as shown in most of figures. In average only the genetic algorithm execution time was in the interval of $(65, 94)s$ with an average value of $81.3s$ which is comparable with other previously tested algorithms such as A^* which showed over the same conditions an average execution time of $83s$.

TABLE III. Testing PC Features.

CPU	AMD Athlon Gold 3150U @ 2.400GHz
cores	2 hardware (4 subprocess)
GPU	AMD ATI 03:00.0 Picasso
RAM	12 GB
main drive	SSD
OS	Ubuntu 20.04.3 LTS x86_64

The genetic algorithm took in average 51.5 generations (iterations) to reach an average $f(X) = 1115$, the Fig. 8 shows a plot of one of the tests done, where the best target function in each generation is plotted. This specific example, reaches the convergence value in 44 generations. For the specific algorithm setup used, never this one reaches the maximum number of iterations (200), always this stops by reaching the maximum number of iterations without improvement the fitness function (20), that is meant the algorithm rapidly reaches a minimum but this is not necesary the global (see Fig. 9).

In order to find the global minimum a new test was done, this time with no limits of number of generations and generations without improvement. As a result this last test gave an execution time of $21min\ 33s$, almost 16 times more than the original tests, and this one reaches a $f(X) = 1027.7$

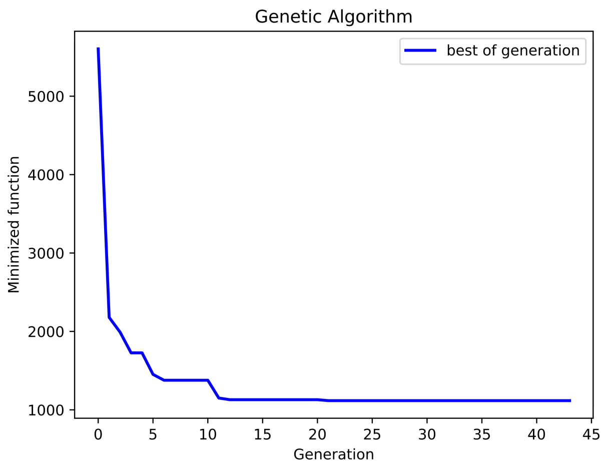

Fig. 8. Target Function Minimizing with a Limit of 20 Generations without Improvements.

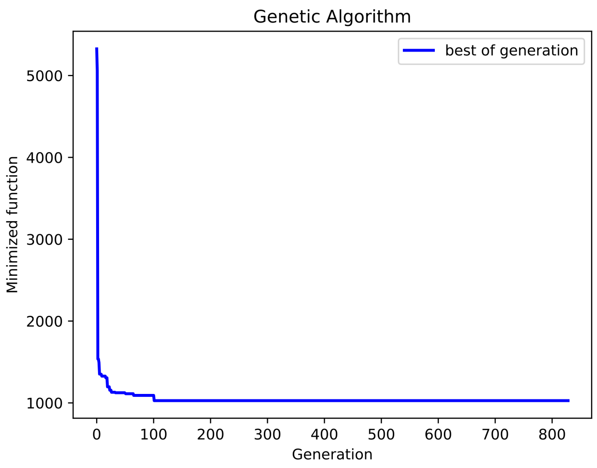

Fig. 10. Target Function Minimizing without a Limit of Generations.

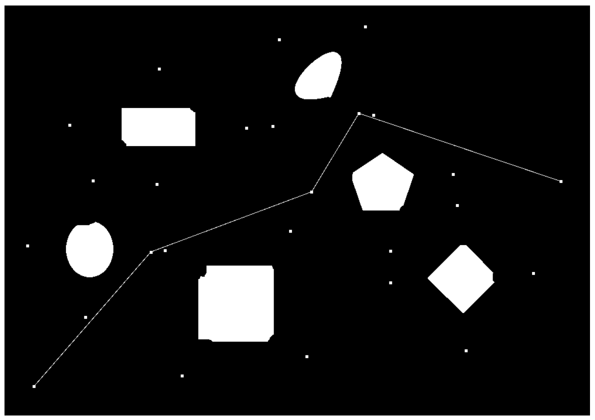

Fig. 9. Obtained Final Path Example.

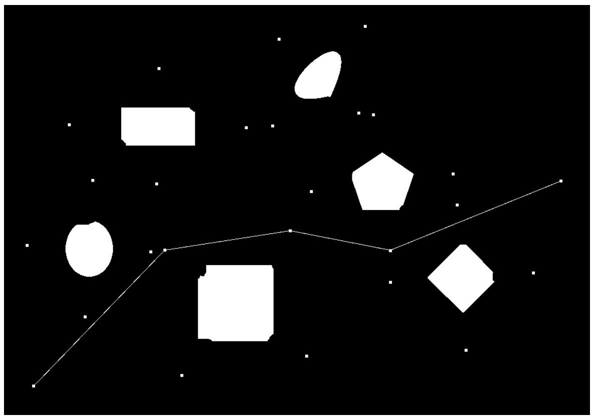

Fig. 11. Obtained Final Path, the Second Example.

which represents an improvement of 7.83%, obtained after 829 generations. The minimizing plot of this second test is shown in Fig. 10, where it is possible to graphically realize that the target function value has practically no changes from the generation number 110. The final path generated by this last test is shown in Fig. 11.

On the other hand, the image processing operations in charge of generating the collision restriction, spend around of the 43% of the total execution time of the genetic algorithm, mainly due to their use of pixel-to-pixel image comparisons which have a high computational cost. This execution time, can be exponentially increased by a linear increasing in the image dimensions.

IV. Conclusion

The proposed strategy of path planning based on visibility graphs and genetic algorithms, gave as a result execution times similar to other previously tested algorithms, in cases of simple environments such as the ones with a low amount of obstacles. According to the optimization parameters and environment images used, the genetic algorithm finds a valid solution with workable execution times. If the resolution of the environment input image or the number of obstacles increases, this proposed strategy will reach high execution times which will make it difficult to apply it on a real time robotics navigation task. This last could happen also is the application needs that the genetic algorithm finds the global minimum.

This proposal implements a safety distance between the obstacles and the mobile robot, this distance is based on an image dilation operation, this technique can limit the possible paths in the visibility graph as shown in Fig. 7 where a connection line is missing for the obstacle in the upper-left corner, as well as obstacles with concave shape could generate issues due to the methodology used for calculating the *keypoints*.

As future work, it is proposed to generate and automatic image resizing (without information losing) in order to reduce its dimensions and therefore the computing time involved in the image operations within the target function to be solved by the genetic algorithm. Another future work proposal is to develop a path planning strategy that uses directly the genetic algorithm over all the navigating free space of the image, so that the solution set will be not an index set but a set of (x, y)

direct points on the image. This last proposal, could take better advantage of the searching features of the genetic algorithms.

Acknowledgment

This work was supported by Universidad Distrital Francisco José de Caldas, specifically by the Technology Faculty. The views expressed in this paper are not necessarily endorsed by Universidad Distrital. The authors thank the ARMOS research group for the simulations and tests done.

References

[1] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[2] R. Manzoor and N. Kumar, "Mobile robot path planning approaches: Recent developments," in *Innovations in Information and Communication Technologies (IICT-2020)*, P. K. Singh, Z. Polkowski, S. Tanwar, S. K. Pandey, G. Matei, and D. Pirvu, Eds. Cham: Springer International Publishing, 2021, pp. 301–308.

[3] A. V. Rendón, "Evaluación de estrategia de navegación autónoma basada en comportamiento reactivo para plataformas robóticas móviles," *Tekhnê*, vol. 12, no. 2, pp. 75–82, 2015.

[4] N. Adzhar, Y. Yusof, and M. A. Ahmad, "A Review on Autonomous Mobile Robot Path Planning Algorithms," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 3, pp. 236–240, 2020.

[5] D. A. R. Ramos, B. A. B. Ruiz, and D. A. M. Osuna, "Desarrollo de control e implementación de robot móvil con la capacidad de solucionar laberintos complejos," *# ashtag*, no. 9, pp. 21–33, 2016.

[6] S. A. Zanlongo, L. Bobadilla, and Y. T. Tan, "Path-planning of miniature rovers for inspection of the hanford high-level waste double shell tanks," in *Florida Conference on Recent Advances in Robotics (FCRAR)*, 2017.

[7] H.-y. Zhang, W.-m. Lin, and A.-x. Chen, "Path planning for the mobile robot: A review," *Symmetry*, vol. 10, no. 10, p. 450, 2018.

[8] F. H. M. Sarmiento and D. A. G. Ramírez, "Openrrarch: una arquitectura abierta, robusta y confiable para el control de robots autónomos," *tecnura*, vol. 21, no. 51, pp. 96–104, 2017.

[9] L. Blasi, E. D'Amato, M. Mattei, and I. Notaro, "Path planning and real-time collision avoidance based on the essential visibility graph," *Applied Sciences*, vol. 10, no. 16, p. 5613, 2020.

[10] N. Roy, R. Chattopadhay, A. Mukherjee, and A. Bhuiya, "Implementation of image processing and reinforcement learning in path planning of mobile robots," *International Journal of Engineering Science*, vol. 15211, 2017.

[11] M. N. Zafar and J. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," *Procedia computer science*, vol. 133, pp. 141–152, 2018.

[12] M. Brand, M. Masuda, N. Wehner, and X.-H. Yu, "Ant colony optimization algorithm for robot path planning," in *2010 international conference on computer design and applications*, vol. 3. IEEE, 2010, pp. V3–436.

[13] K. Akka and F. Khaber, "Mobile robot path planning using an improved ant colony optimization," *International Journal of Advanced Robotic Systems*, vol. 15, no. 3, p. 1729881418774673, 2018.

[14] K. Su, Y. Wang, and X. Hu, "Robot path planning based on random coding particle swarm optimization," *International journal of advanced computer science and applications*, vol. 6, no. 4, pp. 58–64, 2015.

[15] H. Mo and L. Xu, "Research of biogeography particle swarm optimization for robot path planning," *Neurocomputing*, vol. 148, pp. 91–99, 2015.

[16] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Applied Soft Computing*, vol. 59, pp. 68–76, 2017.

[17] X. Li, D. Wu, J. He, M. Bashir, and M. Liping, "An improved method of particle swarm optimization for path planning of mobile robot," *Journal of Control Science and Engineering*, vol. 2020, 2020.

[18] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent bézier curve-based path planning model using chaotic particle swarm optimization algorithm," *Cluster Computing*, vol. 22, no. 2, pp. 4745–4766, 2019.

[19] A. Machmudah, S. Parman, and M. Baharom, "Continuous path planning of kinematically redundant manipulator using particle swarm optimization," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, pp. 207–217, 2018.

[20] E. Dolicanin, I. Fetahovic, E. Tuba, R. Capor-Hrosik, and M. Tuba, "Unmanned combat aerial vehicle path planning by brain storm optimization algorithm," *Studies in Informatics and Control*, vol. 27, no. 1, pp. 15–24, 2018.

[21] A. Pandey and D. R. Parhi, "Optimum path planning of mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm," *Defence Technology*, vol. 13, no. 1, pp. 47–58, 2017.

[22] K. Qian, Y. Liu, L. Tian, and J. Bao, "Robot path planning optimization method based on heuristic multi-directional rapidly-exploring tree," *Computers & Electrical Engineering*, vol. 85, p. 106688, 2020.

[23] M. Radmanesh, M. Kumar, and M. Sarim, "Grey wolf optimization based sense and avoid algorithm in a bayesian framework for multiple uav path planning in an uncertain environment," *Aerospace Science and Technology*, vol. 77, pp. 168–179, 2018.

[24] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018.

[25] R. M. C. Santiago, A. L. De Ocampo, A. T. Ubando, A. A. Bandala, and E. P. Dadios, "Path planning for mobile robots using genetic algorithm and probabilistic roadmap," in *2017IEEE 9th international conference on humanoid, nanotechnology, information technology, communication and control, environment and management (HNICEM)*. IEEE, 2017, pp. 1–5.

[26] X. Liang, P. Jiang, and H. Zhu, "Path planning for unmanned surface vehicle with dubins curve based on ga," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 5149–5154.

[27] C. Ma, R. He, and W. Zhang, "Path optimization of taxi carpooling," *PLoS One*, vol. 13, no. 8, p. e0203221, 2018.

[28] S. Luo, Y. Singh, H. Yang, J. H. Bae, J. E. Dietz, X. Diao, and B.-C. Min, "Image processing and model-based spill coverage path planning for unmanned surface vehicles," in *OCEANS 2019 MTS/IEEE SEATTLE*. IEEE, 2019, pp. 1–9.

[29] F. M. Santa, S. O. Rivera, and M. A. Saavedra, "Enfoque de navegación global para un robot asistente," *Tecnura*, vol. 21, no. 51, p. 105, 2017.

[30] J. Krejsa and S. Vechet, "Determination of optimal local path for mobile robot," in *Mechatronics 2017*, T. Březina and R. Jabłoński, Eds. Cham: Springer International Publishing, 2018, pp. 637–643.

[31] A. Barrero, M. Robayo, and E. Jacinto, "Algoritmo de navegación a bordo en ambientes controlados a partir de procesamiento de imágenes," *Tekhnê*, vol. 12, no. 2, pp. 23–34, 2015.

[32] F. Martinez *et al.*, "Navigable points estimation for mobile robots using binary image skeletonization," in *Eighth International Conference on Graphic and Image Processing (ICGIP 2016)*, vol. 10225. International Society for Optics and Photonics, 2017, p. 1022505.

[33] N. B. A. Latip, R. Omar, and S. K. Debnath, "Optimal path planning using equilateral spaces oriented visibility graph method," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 6, p. 3046, 2017.

[34] J. D. Contreras, F. Martínez *et al.*, "Path planning for mobile robots based on visibility graphs and a* algorithm," in *Seventh International Conference on Digital Image Processing (ICDIP 2015)*, vol. 9631. SPIE, 2015, pp. 345–350.

[35] S. B. Mane and S. Vhanale, "Genetic algorithm approach for obstacle avoidance and path optimization of mobile robot," in *Computing, Communication and Signal Processing*, B. Iyer, S. Nalbalwar, and N. P. Pathak, Eds. Singapore: Springer Singapore, 2019, pp. 649–659.